Using Well-Known Techniques for Classifying User Behavior Profiles

Jose Antonio Iglesias and Agapito Ledezma and Araceli Sanchis

CAOS Group, Department of Computer Science, Carlos III University of Madrid
Av. Universidad, 30, Leganes, Madrid, Spain
Email: {jiglesia;ledezma;masm}@inf.uc3m.es

Abstract: The security of a computer is based on the realization of confidentiality, integrity, and availability. A computer can keep track of computer users to improve the security in the system. However, this does not prevent a user from impersonating another user. If a computer system can model the behavior of the users, it can be very beneficial detecting masqueraders, assisting them or predicting their future actions. In this paper, we present three different methods for classifying the behavior of a computer user. The proposed three methods are: *Bayesian Netwoks*, *Hidden Markov Models* and a method based on Term Weighting (*TFIDF*). These three techniques have been chosen because we want to assess pure statistical techniques (Bayesian Networks) and information-oriented techniques (TFIDF) against a technique that appears to be more adequate (at least in principle) for identifying the behavior of users (HMMs).
Keywords: Agent behavior, user profile models.

1. Introduction

Knowledge about others is very beneficial for coordination, collaboration and adversarial planning. There are new theories which claim that a high percentage of the human brain capacity is used for predicting the future, including the behavior of other humans [15].

In order to act efficiently, agents (software agents, robots or humans) usually try to recognize the behavior of other agents. To achieve this recognition, agents usually need a plan- or behavior-library that contains behaviors which the observed agent is likely to follow. Different techniques have been used in agent modeling in very different areas; for instance, opponent-modeling in soccer domain simulation [12], intelligent user interface [13], and virtual environment for training [21].

The aim of this research is to present and to evaluate different methods which can represent and classify the behavior of different agents in different domains. Also, we will considerer that sequences are very relevant in human skill learning and in high-level problem solving and reasoning and that the actions performed by an agent could be influenced by his past experiences. For example, in a human-computer interaction by commands, the sequentiality of these commands is essential for the result of the interaction and we will study if this aspect is important for recognizing a computer user.

This paper is organized as follows: Section 2 provides a brief overview of the background and related work relevant to this research. In Section 3, the goal of the research is formally explained. The three proposed different methods based on well known techniques are explained in section 4. Section 5 describes the experimental setting and the experimental results obtained in the three different methods. Finally, section 5 contains future work and concluding remarks.

2. Related Work

In many different areas it is very useful to model and classify the behavior of other agents. In competitive domains, the opponent knowledge can be very advantageous. There are many works in which a team is modelled and classified using different methods, such as *Hidden Markov Models* [7], *Deterministic Finite Automatons* [3] and decision trees [17]. Kaminka et al. [10] recognize basic actions based on descriptive predicates, and detect the relevant actions of a team using a statistical approach. However, that work is focused on unsupervised learning, with no ability to classify behaviors into classes.

Different methods have been used to find out relevant information under the human behavior in many different areas. Macedo et al. [14] propose a system (*WebMemex*) that provides recommended information based on the captured history of navigation from a list of known users. Pepyne et al. [16] propose a method using queuing theory and logistic regression modeling methods for profiling computer users based on simple temporal aspects of their behavior. In this case, the goal is to create profiles for very specialized groups of users, who would be expected to use their computers in a very similar way. Gody and Amandi [6] present a technique to generate readable user profiles that accurately capture interests by observing their behavior on the Web. The proposed technique is built on the *Web Document Conceptual Clustering* algorithm, with which profiles without an a priori knowledge of user interest categories can be acquired.

In the computer intrusion detection problem, Coull et al. [4] propose an algorithm that uses pairwise sequence alignment to characterize similarity between sequences of commands. The algorithm produces an effective metric for distinguishing a legitimate user from a masquerader. Schonlau et al. [20] investigate a number of statistical approaches for detecting masqueraders.

3. UNIX User Classification

The goal of UNIX user classification is the recognition of a UNIX user profile from the commands s/he types and the classification of this user into a predefined profile. Therefore, as most of the agent modeling techniques, in the implemented

methods in this research, we classify the UNIX commands into the profiles stored in a library. This classification result can be very useful, for example, in computer intrusion detection.

In this research, the profile of a UNIX user is defined by the commands typed during a period of time. The actions executed by a user being inherently sequential, there are many different behaviors that can be represented by a sequence of events. Therefore, the methods proposed in this research can be used in many other environments (such as GUI events, network packet traffic and so on).

As a behavior is represented by a sequence of elements, the behavior classification can be defined as follows: Let us define a sequence of *n* elements as *E* = $\{e_1, e_2,..., e_n\}$. Given a set of *m* classes C = $\{c_1, c_2,..., c_m\}$ we wish to determine which class $c_i \in$ C the sequence *E* belongs to.

4. Description of the proposed classifiers

In this section we briefly describe three different methods that have been implemented and evaluated for creating UNIX user profiles and classifying them. These novel methods are based on well-known techniques.

4.1 Term Weighting: TFIDF

This kind of classifier is based on the relevance feedback algorithm proposed by Rocchio [18]. TFIDF (Term FrequencyInverse Document Frequency) is a common method often used in Information Retrieval (IR) problems. This method is based on a statistical measure (*weight*) used to evaluate how *important* a word is to a document in a collection: First, it evaluates the word frequency in the document (the more a word appears in the document, the more it is considered to be significant in the document). In addition, IDF measures how frequent a word is in the collection.

There are different variants of *TFIDF*; in this research we consider the following weight, as explained in [19]:

$weight_{wd}TFIDF = tf_{wd} * \log \frac{N_d}{n_w}$

where tf_{wd} is the frequency of a the word *w* in the document *d*, *N* is the number of documents in the collection and n_w is the number of documents in which the word *w* appears.

4.1.1 The proposed TFIDF Classifier

In order to classify a UNIX user, her/his profile must be created in advance. To apply the proposed classifier, a profile based on TFID is created for each UNIX user. The *importance* of each command will be used to identify the user. Therefore, we calculate the *TFIDF* weight for each command of the sequence of a user.

In this case, the *TFIDF* weight is calculated as follows:

$TFIDFweight_{cs}(Useru) = tf_{cs} * \log \frac{N_u}{n_c}$

where tf_{cs} is the frequency of the command *c* in the sequence *s* which belongs to the User *u*, N_u is the number of users and n_c is the number of users who have typed the command *c* at least once.

Once the *importance* (TFIDF weight) of each command is calculated, the model of a UNIX user is represented by the distribution of these values. Each user profile is stored in a library (similar to the plan-libraries used in the plan recognition) where it is labeled with a *name* that identifies them.

After creating all the user profiles, a given sequence of commands is obtained and it needs to be classified. Therefore, given a set of profiles stored in a library and a sequence of commands, the goal is to determine into which profile the given sequence fits. Firstly, the distribution of TFIDF weights is calculated from the sequence to classify. Then, it is matched with all the behavior models stored in the library.

As both models are represented by a distribution of elements, this classifier applies a statistical test for matching these distributions. A non-parametric test (or distribution free) is used because this kind of test does not assume a particular population distribution. The proposed test applied is a modification of *Chi-Square* Test for two samples. The profile to classify is considered as an observed sample and all the profiles stored in the library are considered as expected samples. This test compares the observed distribution with all the expected distributions objectively and evaluates if a deviation appears.

The proposed test is the comparison of two sets of *TFIDF weights* in which *Chi-Square* is the sum of the terms $\frac{(Exp-Obs)^2}{Obs}$. This comparison obtains a value (*comparing value*) that indicates the difference (deviation) between the two distributions. The lower this value, the closer the similarity between the two profiles. This comparison test is applied once for each library profile. The profile which obtains the lowest deviation is considered as the most similar one. An advantage of the proposed test is its rapidity because only the observed subsequences are evaluated. However, there is no penalty for the expected relevant subsequences which do not appear in the observed distribution.

Fig. 1 graphically represents the proposed classifier. In section 6 the results using this method in a real environment are shown. Finally, it is remarkable that with this method, we have not considered that the commands typed by a user are sequential. This aspect will be taken into account in the next two proposed classifiers.

4.2 Graphical Model: Bayesian Networks

A Bayesian network is a graphical model that encodes probabilistic relationships among variables of interest. Over the last decade, the Bayesian network has become a popular representation for encoding uncertain expert knowledge in expert systems [9].

Let *U*=$\{x_1, x_2,..., x_n\}$, n≥1 be a set of variables. A *Bayesian network B* over a set of variables *U* is a *network structure* B_s, which is a directed acyclic graph (DAG) over *U* and a set of probability tables B_p = $\{p(u|pa(u))|u \in U\}$ where *pa*(*u*) is the set of parents of *u* in B_S. A Bayesian network represents a probability distribution *P*(*U*) = $\prod_{u \in U} p(u|pa(u))$.

4.2.1 The proposed Bayesian Network Classifier

Using this technique, the classification task consists of classifying a variable *y* = x_0 (called *class variable*) given a set of variables x = $x_1, x_2,..., x_n$ (called *attribute variables*). In this case, a classifier h : x → y is a function that maps an instance of *x* to value of *y*. The classifier is learned from a dataset D con-

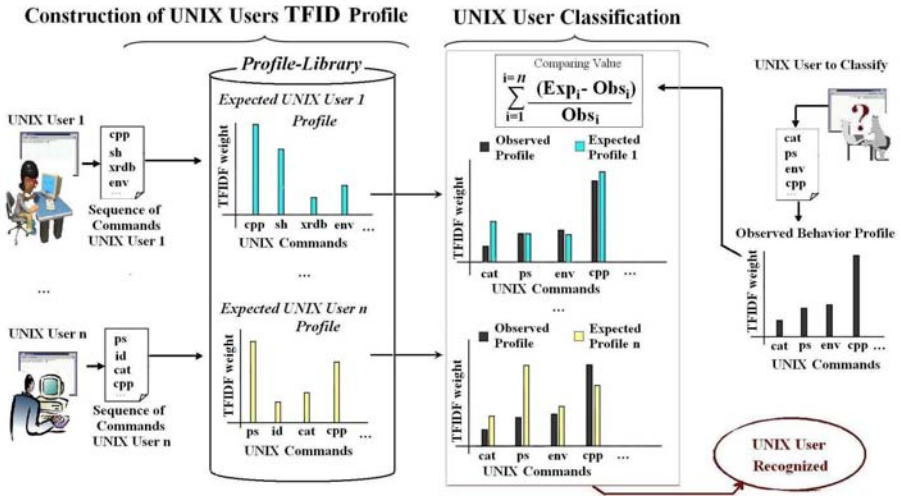

Fig. 1. Framework - The proposed TFIDF classifier

sisting of samples over (x,y). The learning task consists of finding an appropriate Bayesian network given a data set *D* over *U*. Therefore, as is shown in [2], to use a *Bayesian network* as a classifier, the $argmax_y$ *P(y|x)* is calculated using the distribution *P(U)* represented by the Bayesian netwrok. Also, as all variables in *x* are known, complicated inference algorithms are not needed, but just calculate $P(y|x)\prod_{u\in U} p(u|pa(u))$ for all class values.

In this research, in order to create a Bayesian network classifier for UNIX users, the data set consists of sequences of commands that the user have typed consecutively. For obtaining these instances (data set), the sequence of commands is segmented in subsequence of equal length from the first to the last element. Thus, the sequence $A=A_1A_2...A_n$ (where *n* is the number of commands of the sequence) will be segmented in the subsequences described by $A_i...A_{i+length}\ \forall\ i$,*i*=[1,*n-length*+1], where *length* is the size of the subsequences created and this value determines how many commands are considered as dependent.

For example, lets consider the sequence typed by User1 is: *cpp, sh, xrdb, mkpts, env, ps, hostname, id, cat* and each instance consists of 5 attribute variables (subsequences of 5 commands); therefore, the following instances are obtained: {*cpp, sh, xrdb, mkpts, env,* user1}, {*sh, xrdb, mkpts, env, ps,* user1}, {*xrdb, mkpts, env, ps, hostname,* user1}, {*mkpts, env, ps, hostname, id,* user1}, {*env, ps, hostname, id, cat,* user1} where the 5 first words represent the *attribute variables* and the last word represents the *class variable*.

Once the instances are created, the classification can be performed using *WEKA* (Waikato Environment for Knowledge Analysis). *WEKA* is a Java software package [5] with an open source issued under the GNU General Public License. The package provides a collection of machine learning algorithms for data mining tasks. The results of the experiments using this technique are shown in Section 5.

4.3 Hidden Markov Models (*HMM*)

Recent researches have demonstrated the effectiveness of Hidden Markov Models (*HMM*s) for information extraction and they are very used in speech recognition. Also, related with this research, *HMMs* have been used for recognizing automated robot behavior [8].

A *HMM* is a finite state automaton with stochastic state transitions and symbol emissions. At each time step, the *HMM* system (represented by a set of discrete states) is in one state s_i. The state transitions occur according to a certain probability distribution: $Pr(S_{t+1} = s_j \mid S_t = s_i)$ for the state transition $s_i \rightarrow s_j$. However, the system state at time t is not directly observable (*Hidden*). Instead, a set of state dependent observation variables, o_i, are available. For each state s_i, an observation probability $b_i(o)$ is defined over o.

4.3.1 The proposed HMM Classifier

In the proposed *HMM* classifier, the set of Markov states is not previously defined; however, it could correspond to a model of the different *mental states* of the UNIX user. For applying this classifier to identify UNIX users, we need previously to create each UNIX user profile. In this case, each user is represented by a *HMM* with a number of states chosen by the designer (for the experiments conducted in this research, we have used different number of states). Therefore, a *HMM* is created and trained for each user (class). This step requires that the training data (sequence of command) be *labelled* (with the user typed the commands). All the user profiles based on *HMMs* are stored in a library.

Once the profiles have been created, a sequence of commands is given for classifying it. Then, the probability of the sequence is evaluated under each *HMM*. For this process, the *Viterby algorithm* can be used. The *Viterby algorithm* is a dynamic programming algorithm for finding the most likely sequence of hidden states that results in a sequence of observed elements. Finally, the given sequence is classified by the *HMM* (user profile) which gave it the highest likelihood.

This classifier has been implemented by resorting to *UMD HMM : Hidden Markov Model Tool*[11].

5. Experiments

Under the UNIX operating system users type several commands that characterize a user profile. In order to evaluate the

three proposed classifiers, we conducted extensive experiments with real-data of UNIX users.

5.1 UNIX Users Data

In this research, we have used the same data used in the Schonlau et al. [20] masquerade-detection studies (These data are available from the Schonlau web page: http://www.schonlau.net/). These user commands were captured from the UNIX *acctt* auditing mechanism. However, this analysis is only based on two fields: *Command name* and *User*. Thus, a user is identified by a set of commands concatenated by date order; for example the first 10 commands of the *User1* are: *cpp, sh, xrdb, cpp, sh, xrdb, mkpts, env, csh, csh*.

In the command extracting process done by Schonlau et al., the first 15,000 commands for each of about 70 users were recorded over a time period of several moths. Some users generate 15,000 commands in a few days, others in a few months. Some commands recorded by the system are not explicitly typed by the user. For example, a *shell file* contains multiple commands, and running it will cause all of its commands to be recorded.

In [20], Scholen et al. use the data of 50 users randomly selected in which data from the remaining 20 users are interspersed as masqueraders data. However, in this research we obtain the data of the 50 users without data interspersed. For evaluating the different classifiers, we use the sequences of commands typed by 50 different users. In our case, each subsequence contains 12,500 commands.

5.2 Experimental Design

To measure the performance of the 3 classifiers using the above data, the well-established technique of cross-validation is used. For this research, 10-fold cross-validation is chosen. Thus, the 12,500 commands typed by a user (training set) are divided into 10 disjoint subsets with equal size (1,250 commands). Each of the 10 subsets is left out in turn for evaluation.

5.3 Results

In this sections the results obtained using the 3 different classifiers based on different techniques are shown.

5.3.1 TFIDF Classifier Results

In this case, a full program to profile and classify UNIX users using the TFIDF classifier has been implemented. For evaluating the results, each user classification returns a ranking list with the most similar users (*training profile*) to the sequence to classify (*testing profile*). There are users whose behavior is quite similar and the results of the comparison could be similar. However, in these results, a comparison is considered correct only if the user who typed the sequence of commands to classify holds the first position of the ranking list.

Fig. 2 shows the result of the classification of 50 UNIX users using 10-fold cross-validation. Each UNIX user evaluated is represented in the X-axis. The Y-axis represents the percentage of folds that has been correctly classified. Observing this graph, we can see that all the users are correctly classified at least once of the 10 times executed. However the average correct classification percentage value is: **57,1%**.

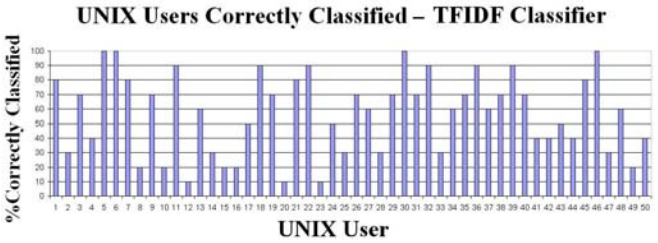

Fig. 2. Classification results - TFIDF classifier

5.3.2 Bayesian Network Classifier Results

In this study, the Bayesian Network classifier has been implemented using WEKA. Also, these results have been conducted using 5 attribute variables. The results have been obtained using WEKA's default settings. In this case, we calculate the percentage of the instances correctly classified using the confusion matrix obtained.

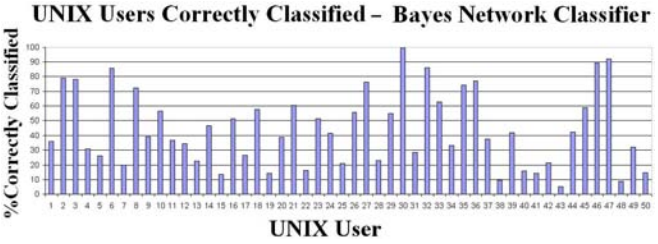

Fig. 3. Classification results - Bayesian network classifier

As in the previous result, Fig. 3 shows the result of the Bayesian Network classification of 50 UNIX users using 10-fold cross-validation. In this case, although we consider that the order of the commands is relevant for classifying a user, the average correct classification percentage value is: **44.2%**.

5.3.3 *HMM* Classifier Results

For creating a system to classify UNIX users using the *HMM* classifier proposed, we have used *UMDHMM : Hidden Markov Model Tool*. In this case, as the result can vary depending of the number of states used for creating the *HMM profile*, we obtain the results using 3, 5 and 10 states. As the TFIDF Classifier, this classifier returns a ranking list with the most similar users to the sequence to classify and only if the correct user holds first position, it is considered a correct classification.

Fig. 4 shows the results of the *HMM* classification. In this case, the results depend of the number of states used for creating the *HMM* profile. The average correct classification percentage value is: **71,8%** (3 States), **73,6%** (5 States) and **76,4%** (10 States). As we can see, this percentage is higher fs the *HMM* profile consists of more states. However, using more states, the *HMM* created is more complex and the process for classifying a user is more time-consumed.

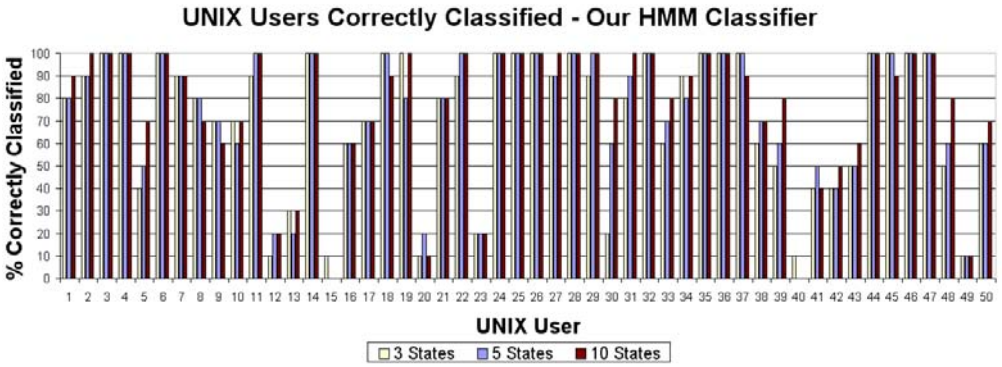

Fig. 4. Classification results - *HMM* classifier

6. Conclusions and Future Work

This paper presents three different methods based on well-known techniques for profiling and classifying UNIX users from the commands they type. The sequence of commands of a user is analyzed using different well known techniques and creating the corresponding profiles. Finally, a given user sequence is classified in the profiles previously created.

The three proposed classifiers have been evaluated with real-data analyzing 50 different UNIX users. A large set of experiments were conducted and the corresponding results are highly significant. These results show that the proposed technique based on *HMMs* is a powerful method for the classification of behavior in this environment, especially if it is compared with a pure statistical technique (Bayesian Networks) and an information-oriented technique (TFIDF).

The results are very encouraging because different actions in the computer system (such as to assist the user, predicting his/her future actions or detecting masqueraders) can be executed. However, the proposed classifiers are generalizable enough to be evaluated in other different domains.

A widely acknowledged challenge in the proposed environment is how to accurately profile a user while s/he changes constantly. For this case, the user profiles should be frequently revised to keep them up to date. This aspect has not been tackled deeply in this research; however, it could be dealt with by using *Evolving Systems* [1] and it is proposed for future work. Finally, it could be very interesting to use the proposed approach for clustering users with similar profiles. This aspect could be easily implemented by comparing the user profiles created with the different proposed classifiers. As a matter of fact, several online learning techniques (including reinforcement learning and ACO) might be used.

Acknowledgments

This work has been supported by the Spanish Ministry of Education and Science under grant TRA-2007-67374-C02-02.

References

[1] P P Angelov, Evolving rule-based models: a tool for design of flexible adaptive systems, Springer-Verlag, London, UK, 2002.
[2] R R Bouckaert, Bayesian networks in weka, Tech. Rep. 14/2004, Computer Science Department. University of Waikato, 2004.
[3] D Carmel and S Markovitch, Opponent modeling in multi-agent systems, In Adaptation and Learning in Multi-Agent Systems. Springer-Verlag: Heidelberg, Germany, 1996, pp. 40–52.
[4] S Coull, J Branch, B Szymanski and E Breimer, Intrusion detection: A bioinformatics approach. ACSAC '03: Proceedings of the 19th Annual Computer Security Applications Conference Washington, DC, USA, 2003, IEEE Computer Society, p. 24.
[5] S Garner, Weka: The waikato environment for knowledge analysis. Proc. of New Zealand Computer Science Research Students Conference 1995, pp. 57–64.
[6] A Godoy D.; Amandi, User profiling for web page filtering. Internet Computing, IEEE, Vol. 9, No. 4, 2005, pp. 56–64.
[7] K Han and M Veloso, Automated robot behavior recognition applied to robotic soccer. IJCAI-99 1999.
[8] K Han and M Veloso, Automated robot behavior recognition applied to robotic soccer, In Robotics Research: the Ninth International Symposium, J Hollerbach and D Koditschek, Eds. Springer-Verlag, London, 2000, pp. 199–204.
[9] D Heckerman, D Geiger and D M Chickering, Learning bayesian networks: The combination of knowledge and statistical data. Machine Learning, Vol. 20, No. 3, 1995, pp. 197–243.
[10] G A Kaminka, M Fidanboylu, A Chang and M M Veloso, Learning the sequential coordinated behavior of teams from observations. RoboCup 2002, Vol. 2752 of LNCS, Springer, pp. 111–125.
[11] T Kanungo, Umdhmm: Hidden markov model toolkit, In Extended Finite State Models of Language, A Kornai, Ed. Cambridge University Press, 1999.
[12] A Ledezma, R Aler, A Sanchs and D Borrajo, Predicting opponent actions by observation.. RoboCup 2004, Vol. 3276 of LNCS, Springer, pp. 286–296.
[13] N Lesh, C Rich and C L Sidner, Using plan recognition in human-computer collaboration. UM99 1999, pp. 23–32.
[14] A A Macedo, K N Truong, J A Camacho-Guerrero and M da GraÇa Pimentel, Automatically sharing web experiences through a hyperdocument recommender system. HYPERTEXT 2003 New York, NY, USA, 2003, ACM, pp. 48–56.
[15] N J Mulcahy and J Call, Apes save tools for future use. Science, Vol. 312, No. 5776, 2006, pp. 1038–1040.
[16] H J G W Pepyne D.L., User profiling for computer security. American Control Conference 2004, pp. 982–987.
[17] P Riley and M M Veloso, On behavior classification in adversarial environments. DARS 2000, pp. 371–380.
[18] J Rocchio, Relevance feedback in information retrieval. The SMART retrieval system: Experiments in Automatic Document Processing 1971, LNCS, Prentice Hall, pp. 313–323.
[19] G Salton, Automatic Text Processing. The Transformation, Analysis, and Retrieval of Information by Computer, Addison–Wesley, 1989.
[20] M Schonlau, W DuMouchel, W Ju, A Karr, M Theus and Y Vardi, Computer intrusion: Detecting masquerades. Statistical Science, Vol. 16, No. 1, 2001, pp. 58–74.
[21] M Tambe and P S Rosenbloom, Resc: An approach for dynamic, real-time agent tracking. IJCAI-95 Montreal, Canada, 1995.